Using Locally Weighted Regression to Estimate the Functional Size of Software: a Preliminary Study

Luigi Lavazza Angela Locoro
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
email:{luigi.lavazza, angela.locoro}@uninsubria.it

Roberto Meli
DPO
Rome, Italy
email:roberto.meli@dpo.it

***Abstract*—In software engineering, measuring software functional size via the IFPUG (International Function Point Users Group) Function Point Analysis using the standard manual process can be a long and expensive activity. To solve this problem, several early estimation methods have been proposed and have become *de facto* standard processes. Among these, a prominent one is High-level Function Point Analysis. Recently, the Simple Function Point method has been released by IFPUG; although it is a proper measurement method, it has a great level of convertibility to traditional Function Points and may be used as an estimation method. Both High-level Function Point Analysis and Simple Function Point skip the difficult and time-consuming activities needed to weight data and transaction functions. This makes the process faster and cheaper, but yields approximate measures. The accuracy of the mentioned method has been evaluated, also via large-scale empirical studies, showing that the yielded approximate measures are sufficiently accurate for practical usage. In this paper, locally weighted regression is applied to the problem outlined above. This empirical study shows that estimates obtained via locally weighted regression are more accurate than those obtained via High-level Function Point Analysis, but are not substantially better than those yielded by alternative estimation methods using linear regression. The Simple Function Point method appears to yield measures that are well correlated with those obtained via standard measurement. In conclusion, locally weighted regression appears to be effective and accurate enough for estimating software functional size.**

Keywords–Function Point Analysis; Early Size Estimation; High-level FPA; Simple Function Points; LOcally Estimated Scatterplot Smoothing (LOESS)

I. INTRODUCTION

In the late seventies, Allan Albrecht introduced Function Points Analysis (FPA) at IBM [1], as a means to measure the functional size of software, with special reference to the "functional content" delivered by software providers. Albrecht aimed at defining a measure that might be correlated to the value of software from the perspective of a user, and could also be useful to assess the cost of developing software applications, based on functional user requirements.

FPA is a Functional Size Measurement Method (FSMM), compliant with the ISO/IEC 14143 standard, for measuring the size of a software application in the early stages of a project, generally before actual development starts. Accordingly, software size measures expressed in Function Points (FP) are often used for cost estimation.

The International Function Points User Group (IFPUG) is an association that keeps FPA up to date, publishes the official FP counting manual [2], and certifies professional FP counters. Unfortunately, in some conditions, performing the standard IFPUG measurement process may be too long and expensive, with respect to management needs, because standard FP measurement can be performed only when relatively complete and detailed requirements specifications are available, while functional measures could be needed much earlier for management purposes.

Many methods were invented and used to provide *estimates* of functional size measures, based on fewer or coarser-grained information than required by standard FPA. These methods are applied very early in software projects, even before deciding what process (e.g., agile or waterfall) will be used. One of these methods is the High-level FPA (HLFPA) method [3], which was developed by NESMA under the name of "NESMA estimated" method [4].

In 2010, a new FSMM called Simple Function Point (SiFP) was developed by Meli [5]. In 2019, IFPUG acquired the method and in 2021 the IFPUG branded Simple Function Point (SFP) method was delivered to the market [6].

HLFPA and SiFP have been evaluated by several studies, which found that the methods is usable in practice to approximate traditional FPA values, since they yield reasonably accurate estimates. However, the question if it is possible to get more accurate estimates from the basic information used by HLFPA remains open.

In this paper, we evaluate—via an empirical study—the usage of LOESS (LOcally Estimated Scatterplot Smoothing)—also known as LOWESS (LOcally WEighted Scatterplot Smoothing)—to build models that can be used for early estimation of functional size.

We also compare the standard IFPUG FPA measures, the estimates obtained via HLFPA and the estimates obtained via alternative methods (linear regression models and LOESS models) with the measures obtained via the Simple Function Point (SFP) method. SFP is a lightweight method that has also been adopted by IFPUG as an alternative to full-fledged FPA. SFP measurement requires even less time and effort than HLFPA, and it usually yields measures that are very well correlated with IFPUG standard measures.

The remainder of the paper is organized as follows. Section II provides an overview of functional size measurement methods, and other background information. Section III describes the empirical study and its results. In Section IV, we discuss the threats to the validity of the study. Section V reports about related work. Finally, in Section VI, we draw some conclusions and outline future work.

II. BACKGROUND

Function Point Analysis was originally introduced by Albrecht to measure the size of data-processing systems from the point of view of end-users, with the goal of the estimating value of an application and the development effort [1]. The critical fortunes of this measure led to the creation of the IFPUG (International Function Points User Group), which maintains the method and certifies professional measurers.

The "amount of functionality" released to the user can be evaluated by taking into account 1) the data used by the application to provide the required functions, and 2) the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Both data and transactions are counted on the basis of Functional User Requirements (FURs) specifications, and constitute the IFPUG Function Points measure.

FURs are modeled as a set of base functional components (BFCs), which are the measurable elements of FURs: each of the identified BFCs is measured, and the size of the application is obtained as the sum of the sizes of BFCs. IFPUG BFCs are: data functions (also known as logical files), which are classified into internal logical files (ILF) and external interface files (EIF); and elementary processes (EP)—also known as transaction functions—which are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ), according to the activities carried out within the considered process and the primary intent.

The complexity of a data function (ILF or EIF) depends on the RETs (Record Element Types), which indicate how many types of variations (e.g., sub-classes, in object-oriented terms) exist per logical data file, and DETs (Data Element Types), which indicate how many types of elementary information (e.g., attributes, in object-oriented terms) are contained in the given logical data file.

The complexity of a transaction depends on the number of FTRs—i.e., the number of File Types Referenced while performing the required operation—and the number of DETs—i.e., the number of types of elementary data—that the considered transaction sends and receives across the boundaries of the application. Details concerning the determination of complexity can be found in the official documentation [2].

The core of FPA involves three main activities:

1) Identifying data and transaction functions.
2) Classifying data functions as ILF or EIF and transactions as EI, EO or EQ.
3) Determining the complexity of each data or transaction function.

The first two of these activities can be carried out even if the FURs have not yet been fully detailed. On the contrary, activity 3 requires that all details are available, so that FP measurers can determine the number of RET or FTR and DET involved in every function. Activity 3 is relatively time- and effort-consuming [7].

HLFPA does not require activity 3, thus allowing for size estimation when FURs are not fully detailed: it only requires that the complete sets of data and transaction functions are identified and classified.

The SFP method [6] does not require activities 2 and 3: it only requires that the complete sets of data and transaction functions are identified.

Both the HLFPA and SFP methods let measurers skip the most time- and effort-consuming activity, thus both are relatively fast and cheap. The SFP method does not even require classification, making size estimation even faster and less subjective (since different measurers can sometimes classify differently the same transaction, based on the subjective perception of the transaction's primary intent).

A. The High-level FPA method

NESMA defined two size estimation methods: the 'NESMA Indicative' and the 'NESMA Estimated' methods. IFPUG adopted these methods as early function point analysis methods, under the names of 'Indicative FPA' and 'High-level FPA,' respectively [3]. The Indicative FPA method proved definitely less accurate [8], [9]. Hence, in this paper, we consider only the High-level FPA method.

The High-level FPA method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of functions: ILF and EIF are assumed to be of low complexity, while EI, EQ and EO are assumed to be of average complexity. Hence, estimated size is computed as follows:

$$EstSize_{UFP} = 7\ \#ILF + 5\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ$$

where *#ILF* is the number of data functions of type ILF, *#EI* is the number of transaction functions of type EI, etc.

B. The Simple Function Point Method

The Simple Function Point measurement method [5] [6] has been specifically designed to be agile, fast, lightweight, easy to use, and with minimal impact on software development processes. It is easy to learn and provides reliable, repeatable, and objective results. Like IFPUG FPA, it is independent of the technologies used and technical design principles.

SFP requires only the identification of Elementary Processes (EP) and Logical Files (LF), based on the following assumptions: 1) a user gives value to a BFC as a whole independently of internal organization and details, and 2) a cost model based on SFP shows a precision that is comparable to that of a cost model based on a detailed FPA measure. The latter assumption has been verified by different studies [10] [11].

SFP assigns a numeric value directly to these BFCs:

$$SFP = 7\ \#LF + 4.6\ \#EP$$

thus significantly speeding up the functional sizing process, at the expense of ignoring the domain data model, and the primary intent of each Elementary Process.

The weights for each BFC were originally given to achieve the best possible approximation of FPA but as long as the method has become a measurement method, those weights became constants, which are not subject to update or change for approximation reasons and that are crystallized for stability, repeatability and comparability reasons. We can approximate the FPA by setting $EstSize_{UFP} = SFP$.

III. Empirical study

In the empirical study, we use an ISBSG dataset [12], which was also used previously to evaluate SFP [10].

The ISBSG dataset contains several small project data. As a matter of fact, estimating the size of small projects is not very interesting. A certified function point consultant that performs FP analysis according to the IFPUG standard counts between 400 and 600 FP per day, according to Capers Jones [13] and between 200 and 300 FP per day according to experts from Total Metrics [14]. Therefore, there is hardly any need for estimating the size of projects smaller than 200 UFP, since those projects can be sized accurately in no more than one working day.

Based on these considerations, we removed from the dataset the projects smaller than 200 UFP. The resulting dataset includes data from 110 projects having size in the [207, 4202] range. Some descriptive statistics for this dataset are given in Table I.

TABLE I
Descriptive statistics for the ISBSG dataset.

	UFP	HLFPA	SFP	*#EI*	*#EO*	*#EQ*	*#ILF*	*#EIF*	*#LF*	*#EP*
Mean	976	888	971	43	46	46	26	24	50	135
StDev	842	739	785	38	71	51	22	23	39	123
Median	639	607	674	29	17	32	20	18	37	82
Min	207	202	223	0	0	0	0	1	12	14
Max	4202	3755	4257	204	442	366	100	172	234	656

A. Method used

We build models of functional size using LOESS (locally estimated scatterplot smoothing) [15]. LOESS is a non-parametric regression method that combines multiple regression models in a k-nearest-neighbor-based meta-model. It fits simple models to localized subsets of the data to build up a function that describes the deterministic part of the variation in the data, point by point.

The analysis was carried out using the R programming language and environment [16]. Specifically, we used the `loess` function from the `Stats` package, which is provided as part of the system libraries.

Through the `span` parameter, the `loess` function makes it possible to control the degree of smoothing. In the empirical study, we tried different values for the `span` parameter.

We aimed at building models using the same five variables (*#EI*, *#EO*, *#EQ*, *#ILF*, *#EIF*) used by HLFPA. However, the `loess` function from the `Stats` package does not allow more than 4 independent variable. To overcome this problem, we observe that in the HLFPA method, *#EI* and *#EQ* get the same weight; therefore, it is conceivable to consider EIs and EQs as a single class of transactions (only as far as size estimation is concerned). Accordingly, for each project we compute $\#EIQ = \#EI + \#EQ$. Then we use four independent variables (*#EO*, *#EIQ*, *#ILF*, *#EIF*) to build size models via LOESS. In addition, we built models that use the same two variables (*#LF* and *#EP*) used by SFP. We also built Ordinary Least Square (OLS) linear regression models.

The evaluation was carried out via 10-time 10-fold cross validation. For all the estimates obtained from 10-time 10-fold cross validation, we compute estimation errors and a few indicators, as follows. The error (alias residual) for the i^{th} estimation is defined as $ee_i = S_i - E_i$, where S_i is the actual size of the element involved in the i^{th} estimation (i.e., the size measured according to the IFPUG standard process) and E_i is the estimated size. The computed indicators are:

- MAR is the Mean of Absolute Residuals, i.e., $MAR = \frac{1}{n}\sum_{i=0}^{n} |ee_i|$, where n is the number of estimates.
- MAR/MS is the MAR divided by the mean size $MS = \frac{1}{n}\sum_{i=0}^{n} S_i$. It gives an idea of the relative importance or the estimation errors.
- MMRE is the mean magnitude of relative errors. $MMRE = \frac{1}{n}\sum_{i=0}^{n} |re_i|$, since a relative error is defined as $re_i = \frac{ee_i}{S_i}$. MMRE has been widely criticized as a biased metric [17]: we report it for completeness. At any rate, we also report MAR/MS, which is not a biased metric, since the mean size is a characteristic of the given dataset: MAR/MS is a sort of normalization of the MAR.
- MdMRE is the median magnitude of relative errors.

- Finally, R^2 (the coefficient of determination) is given, since it is a quite reliable indicator of the models' accuracy [18].

B. Results obtained

We carried out 10-times 10-fold cross validation. In the process, we did not always get usable results. Specifically, via OLS regression we sometimes obtained invalid models (e.g., models with not normally distributed residuals); via LOESS we obtained models that did not support estimation in extreme cases, i.e., for too large or too small independent variables. All these cases were not evaluated. They are a strict minority, hence the reported results represent the most likely outcome of estimation in practice.

The accuracy indicators computed over the obtained estimates are given in Table II. Models LMv are built using OLS regression using v independent variables; models LWMv (where LWM stands for Locally Weighted Model) are built using LOESS, based on v independent variables. For LWMv we give in parentheses the value of the span value.

TABLE II
ESTIMATION ACCURACY INDICATORS.

	MAR	MAR/MS	MMRE	MdMRE	R^2
HLFPA	103.8	0.106	0.097	0.084	0.966
LM5	62.0	0.064	0.074	0.057	0.985
LM4	58.2	0.060	0.071	0.055	0.987
LM2	91.6	0.096	0.096	0.084	0.971
LWM4(0.5)	93.7	0.107	0.109	0.089	0.943
LWM2(0.5)	91.4	0.099	0.103	0.082	0.940
LWM4(0.75)	66.5	0.076	0.082	0.068	0.972
LWM2(0.75)	88.7	0.096	0.101	0.075	0.950
LWM4(0.95)	55.6	0.064	0.073	0.064	0.984
LWM2(0.95)	86.6	0.094	0.096	0.072	0.958

Table II suggests that OLS linear models provide quite good estimates. Surprisingly, LM4, i.e., the model based on *#EO*, *#EIQ*, *#ILF*, *#EIF* achieves better results than the LM5, i.e., the model based on *#EO*, *#EI*, *#EQ*, *#ILF*, *#EIF*.

We can also observe that estimation accuracy of LWM models varies with the `span`; specifically, accuracy improves with `span`. However, the improvement is modest for LWM2 (MAR decreases from 91.4 to 86.6), while it is quite large for LWM4 (MAR decreases from 93.7 to 55.6). Overall, it seems that when LOESS is used with two variables it is not able to substantially improve the estimates provided by LM2; instead, LOESS used with four variables achieves good results, provided that `span` is sufficiently large. In fact, the minimum MAR is achieved by LWM4 with `span`=0.95.

To evaluate if the estimates provided by a method are significantly better than those provided by another method, we tested the statistical significance of the differences among absolute errors yielded by the considered methods [17]. Namely, we compared the absolute residuals via Wilcoxon sign rank test [19] (using the `wilcox.test` function from the R `Stats` package). The results (which are all statistically significant at the usual $\alpha = 0.05$ level) are given in Table III.

TABLE III
COMPARISON OF MODEL'S ABSOLUTE RESIDUALS VIA WILCOXON SIGN RANK TEST.

	HLFPA	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	–	>	>	>	>	>	>	>	>	>
LM5	<	–	>	<	<	<	<	<	>	<
LM4	<	<	–	<	<	<	<	<	<	<
LM2	<	>	>	–	<	<	>	=	>	>
LWM4(0.5)	<	>	>	>	–	=	>	>	>	>
LWM2(0.5)	<	>	>	>	=	–	>	>	>	>
LWM4(0.75)	<	>	>	<	<	<	–	<	>	<
LWM2(0.75)	<	>	>	=	<	<	>	–	>	>
LWM4(0.95)	<	<	>	<	<	<	<	<	–	<
LWM2(0.95)	<	>	>	<	<	<	>	<	>	–

To assess the effect size, we use the non-parametric statistic A by Vargha and Delaney [20], as provided by the R package `effsize` [21]. We obtained the results given in Table IV, where each numeric result is accompanied by its interpretation [21]: 'n' and 's' indicate negligible and small effect size, respectively.

LWM4(0.95) appears to be the best model according to MAR (Table II). However, According to the Wilcoxon sign rank test, LM4 is the most accurate model. The disagreement between this two indications is explained by Vargha and Delaney's A, which is 0.51 for LM4 vs. LWM4(0.95), showing that the size effect is practically nil, i.e., LM4 is better, but by a practically irrelevant extent.

Finally, we look into the error distributions yielded by the estimation methods that we used in the study.

Figure 1 shows the boxplots of estimation errors for each of the used methods. It can be noticed that LWM2 models provide exceedingly large errors in a few cases.

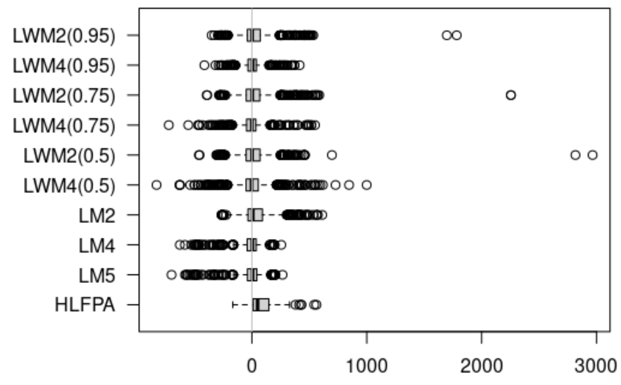

Fig. 1. Error boxplots.

Figure 2 provides the same information as Figure 1, but omitting outliers. It can be seen that the various models do not yield dramatically different accuracy levels, when the outliers are excluded. However, it is noteworthy that HLFPA tends to underestimate (as already noted in [22]). The other models provide more balanced errors, with medians very close to zero.

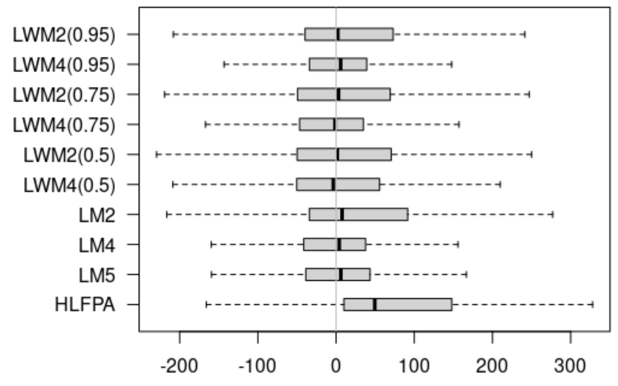

Fig. 2. Error boxplots (no outliers).

Figure 3 shows the boxplots of absolute estimation errors for each of the used methods, excluding outliers. The mean absolute error (i.e., the MAR) is shown as an orange diamond. Also according to Figure 3, LM4, LM5 and LWM4(0.95) are the most accurate models.

Figure 4 shows the distribution of the distance between SFP and IFPUG measures, in comparison with HLFPA and the best estimators. It can be seen that SFP measures provide an approximation that is better than HLFPA's, and not much worse than the best estimators'.

Considering that SFP uses fixed weights and does not even require classifying data and transactions, and that the method is not specifically intended to approximate IFPUG measures, this is a quite remarkable result.

IV. THREATS TO VALIDITY

A typical concern in this kind of studies is the generalizability of results outside the scope and context of the analyzed dataset. In our case, the ISBSG

TABLE IV
EFFECT SIZE ACCORDING TO VARGHA AND DELANEY'S A.

	HLFPA	LM5	LM4	LM2	LWM4(0.5)	LWM2(0.5)	LWM4(0.75)	LWM2(0.75)	LWM4(0.95)	LWM2(0.95)
HLFPA	–	0.61(s)	0.62(s)	0.54(n)	0.53(n)	0.53(n)	0.59(s)	0.55(n)	0.61(s)	0.56(n)
LM5	0.39(s)	–	0.52(n)	0.44(n)	0.42(s)	0.42(s)	0.49(n)	0.43(n)	0.51(n)	0.45(n)
LM4	0.38(s)	0.48(n)	–	0.42(s)	0.40(s)	0.40(s)	0.47(n)	0.42(s)	0.49(n)	0.43(n)
LM2	0.46(n)	0.56(n)	0.58(s)	–	0.48(n)	0.49(n)	0.55(n)	0.50(n)	0.57(n)	0.51(n)
LWM4(0.5)	0.47(n)	0.58(s)	0.60(s)	0.52(n)	–	0.50(n)	0.57(n)	0.52(n)	0.59(s)	0.53(n)
LWM2(0.5)	0.47(n)	0.58(s)	0.60(s)	0.51(n)	0.50(n)	–	0.56(n)	0.51(n)	0.58(s)	0.52(n)
LWM4(0.75)	0.41(s)	0.51(n)	0.53(n)	0.45(n)	0.43(n)	0.44(n)	–	0.45(n)	0.52(n)	0.47(n)
LWM2(0.75)	0.45(n)	0.57(n)	0.58(s)	0.50(n)	0.48(n)	0.49(n)	0.55(n)	–	0.57(n)	0.51(n)
LWM4(0.95)	0.39(s)	0.49(n)	0.51(n)	0.43(n)	0.41(s)	0.42(s)	0.48(n)	0.43(n)	–	0.45(n)
LWM2(0.95)	0.44(n)	0.55(n)	0.57(n)	0.49(n)	0.47(n)	0.48(n)	0.53(n)	0.49(n)	0.55(n)	–

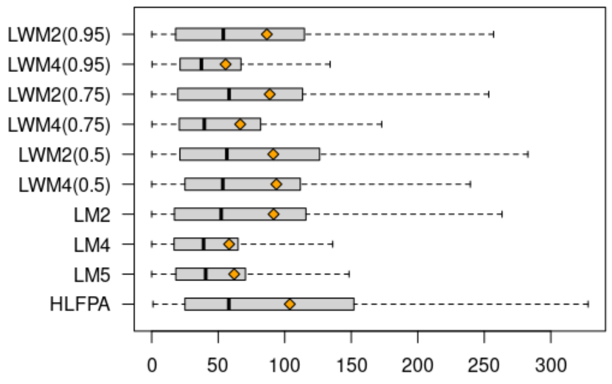

Fig. 3. Absolute error boxplots (no outliers).

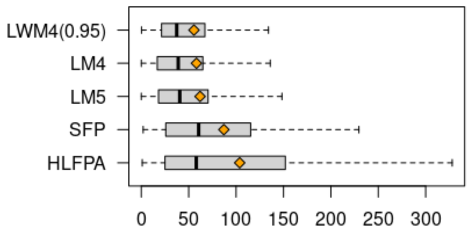

Fig. 4. Distributions of distances from IFPUG measures.

dataset is deemed the standard benchmark among the community, and it includes data from several application domains. Therefore our results may be valid in general.

The usage of MMRE is questionable, since it is has been shown to be a biased indicator (see for instance [17]). Nonetheless, we used MMRE together with other indicators—like MAR, the boxplots of residuals and R^2—to provide a more complete and balanced picture of the accuracy of our results, and compared the precision of different models via sound statistical tests, namely Wilcoxon sign rank test and Vargha and Delaney's A measure of effect size. Therefore, the role of MMRE in the presented evaluations is marginal.

V. RELATED WORK

The quest for measures that are available in the early stages of the software lifecycle dates back to decades ago [23] [24] [25].

The "Early & Quick Function Point" (EQFP) method [26] uses analogy (similarities between a new and a classified piece of software) and analysis (statistical analysis of the estimated similarity) to get size estimates. It was reported that estimates are within $\pm 10\%$ of the real size in most real cases, while the savings in time and costs are between 50% and 90%.

"Easy Function Points," [27], adopt probabilistic approaches to estimate not only the size, but also the probability that the actual size is equal to the estimate.

Lavazza et al. built estimation models for UFP based on BFCs [28] using Least Median Squares robust regression models. They observed that FP measures could be altogether replaced by measured based on a smaller set of BFCs.

Several other early estimation methods were proposed: Table V list the most popular ones.

TABLE V
EARLY ESTIMATION METHODS: DEFINITIONS AND EVALUATIONS

Method name	Definition	Used functions	Weight	Evaluation
NESMA indicative	[29] [30]	data	fixed	[4] [31]–[35] [9]
NESMA estimated	[29] [30]	all functions	fixed	[4] [31]–[35] [9]
Early & Quick FP	[25] [36] [26]	all functions	statistics	[9] [37]
Tichenor ILF model	[38]	ILF	fixed	[9]
simplified FP (sFP)	[39]	all functions	fixed	[9]
ISBSG average weights	[40]	all functions	statistics	[9]
SiFP	[5]	data and trans.	statistics	[10] [11]

Lavazza and Liu [22] used a dataset containing data from 479 projects to compare the accuracy of HLFPA method with Ordinary Least Squares method, with both 5 predictors (LM5) and only 2 predictors (LM2). They found that (1) unlike HLFPA, linear regression models do not underestimate, (2) linear regression models yield slightly less accurate estimates, and (3) models based on only two variables yield marginally less accurate estimates.

VI. CONCLUSION

Measuring software functional size via IFPUG FPA with the standard manual process is sometimes a long and expensive activity, and it is simply impossible when the details of a functional specification are not available for any reason. To solve this problem, several early estimation methods have been proposed. In this paper, we compare the estimates obtained via a standard estimation methods, namely HLFPA, and a new functional size measurement method, namely IFPUG SFP, with the estimates obtained with traditional (namely, linear regression) models and LOESS models. The accuracy achieved by these methods has been evaluated via an empirical study, which used a dataset containing data from 110 projects.

LOESS provided the lowest mean absolute error. However, statistical tests show that linear regression models using 4 or 5 independent variables achieve the same level of accuracy. Therefore, practitioners needing to estimate software functional size in the early stages of projects are advised to try both linear regression models and LOESS models.

ACKNOWLEDGMENT

The work reported here was partly supported by Fondo per la Ricerca di Ateneo, Università degli Studi dell'Insubria.

REFERENCES

[1] A. J. Albrecht, "Measuring application development productivity," in Proceedings of the joint SHARE/GUIDE/IBM application development symposium, vol. 10, 1979, pp. 83–92.
[2] International Function Point Users Group (IFPUG), "Function point counting practices manual, release 4.3.1," 2010.
[3] A. Timp, "uTip – Early Function Point Analysis and Consistent Cost Estimating," 2015, uTip # 03 – (version # 1.0 2015/07/01).

[4] H. van Heeringen, E. van Gorp, and T. Prins, “Functional size measurement-accuracy versus costs–is it really worth it?” in Software Measurement European Forum (SMEF), 2009.

[5] R. Meli, “Simple function point: a new functional size measurement method fully compliant with IFPUG 4.x,” in Software Measurement European Forum, 2011.

[6] IFPUG, “Simple Function Point (SFP) Counting Practices Manual Release 2.1,” 2021.

[7] L. Lavazza, “On the effort required by function point measurement phases,” International Journal on Advances in Software, vol. 10, no. 1 & 2, 2017.

[8] nesma, “Early Function Point Analysis,” https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/ last access 6/6/22.

[9] L. Lavazza and G. Liu, “An empirical evaluation of simplified function point measurement processes,” Journal on Advances in Software, vol. 6, no. 1& 2, 2013.

[10] L. Lavazza and R. Meli, “An evaluation of simple function point as a replacement of IFPUG function point,” in IWSM–MENSURA 2014. IEEE, 2014, pp. 196–206.

[11] F. Ferrucci, C. Gravino, and L. Lavazza, “Simple function points for effort estimation: a further assessment,” in 31st Annual ACM Symposium on Applied Computing. ACM, 2016, pp. 1428–1433.

[12] International Software Benchmarking Standards Group, ““Worldwide Software Development: The Benchmark, release 11,” ISBSG, 2009.

[13] C. Jones, “A new business model for function point metrics,” 2008, http://concepts.gilb.com/dl185 last access 6/6/22.

[14] Total Metrics, “Methods for Software Sizing – How to Decide which Method to Use,” 2007 last access 6/6/22, https://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf.

[15] W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” Journal of the American statistical association, vol. 74, no. 368, 1979, pp. 829–836.

[16] R core team, “R: a language and environment for statistical computing,” 2015.

[17] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd, “What accuracy statistics really measure [software estimation],” in Software, IEE Proceedings-, vol. 148, no. 3. IET, 2001, pp. 81–85.

[18] D. Chicco, M. J. Warrens, and G. Jurman, “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” PeerJ Computer Science, vol. 7, 2021, p. e623.

[19] J. Cohen, “Statistical power analysis for the behavioral sciences Lawrence Earlbaum Associates,” Hillsdale, NJ, 1988, pp. 20–26.

[20] A. Vargha and H. D. Delaney, “A critique and improvement of the cl common language effect size statistics of mcgraw and wong,” Journal of Educational and Behavioral Statistics, vol. 25, no. 2, 2000, pp. 101–132.

[21] M. Torchiano et al., “effsize: Efficient effect size computation,” R package version 0.7, vol. 1, 2017.

[22] G. Liu and L. Lavazza, “Early and quick function points analysis: Evaluations and proposals,” Journal of Systems and Software, vol. 174, 2021, p. 110888.

[23] D. B. Bock and R. Klepper, “FP-S: a simplified function point counting method,” Journal of Systems and Software, vol. 18, no. 3, 1992, pp. 245–254.

[24] G. Horgan, S. Khaddaj, and P. Forte, “Construction of an FPA-type metric for early lifecycle estimation,” Information and Software Technology, vol. 40, no. 8, 1998, pp. 409–415.

[25] L. Santillo, M. Conte, and R. Meli, “Early & Quick Function Point: sizing more with less,” in 11th IEEE International Software Metrics Symposium (METRICS’05). IEEE, 2005, pp. 41–41.

[26] DPO, “Early & Quick Function Points Reference Manual - IFPUG version,” DPO, Roma, Italy, Tech. Rep. EQ&FP-IFPUG-31-RM-11-EN-P, April 2012.

[27] L. Santillo, “Easy Function Points – ‘Smart’ Approximation Technique for the IFPUG and COSMIC Methods,” in IWSM–MENSURA, 2012.

[28] L. Lavazza, S. Morasca, and G. Robiolo, “Towards a simplified definition of function points,” Information and Software Technology, vol. 55, no. 10, 2013, pp. 1796–1809.

[29] NESMA–the Netherlands Software Metrics Association, “Definitions and counting guidelines for the application of function point analysis. NESMA Functional Size Measurement method compliant to ISO/IEC 24570 version 2.1,” 2004.

[30] International Standards Organisation, “ISO/IEC 24570:2005 – Software Engineering – NESMA functional size measurement method version 2.1 – definitions and counting guidelines for the application of Function Point Analysis,” 2005.

[31] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, and N. Lester, “The value of software sizing,” Information and Software Technology, vol. 53, no. 11, 2011, pp. 1236–1249.

[32] J. Popović and D. Bojić, “A comparative evaluation of effort estimation methods in the software life cycle,” Computer Science and Information Systems, vol. 9, no. 1, 2012, pp. 455–484.

[33] P. Morrow, F. G. Wilkie, and I. McChesney, “Function point analysis using nesma: simplifying the sizing without simplifying the size,” Software Quality Journal, vol. 22, no. 4, 2014, pp. 611–660.

[34] L. Lavazza and G. Liu, “An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates,” in ICSEA, 2019, pp. 24–29.

[35] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, “Assessing the effectiveness of approximate functional sizing approaches for effort estimation,” Information and Software Technology, vol. 123, July 2020.

[36] T. Iorio, R. Meli, and F. Perna, “Early&quick function points® v3. 0: enhancements for a publicly available method,” in SMEF, 2007, pp. 179–198.

[37] R. Meli, “Early & quick function point method-an empirical validation experiment,” in Int. Conf. on Advances and Trends in Software Engineering, Barcelona, Spain, 2015.

[38] C. Tichenor, “The IRS development and application of the internal logical file model to estimate function point counts,” in IFPUG Fall Conf., 1997.

[39] L. Bernstein and C. M. Yuhas, Trustworthy systems through quantitative software engineering. John Wiley & Sons, 2005, vol. 1.

[40] R. Meli and L. Santillo, “Function point estimation methods: A comparative overview,” in FESMA, vol. 99. Citeseer, 1999, pp. 6–8.